02

Introducción a orientación de objetos

Transcripción

[00:00] Iniciando nuestro curso de introducción a objetos, vamos a comprender un poco de dónde sale esta inspiración de la orientación a objetos, cómo es que vemos la necesidad de poco a poco abstraer cada parte de nuestro sistema. Antes, digamos hace un tiempo, el tipo de desarrollo de software no era como lo conocemos hoy en día.

[00:26] Por ejemplo un equipo de desarrollo de software está compuesto por varias personas, cada uno encargado de una parte del sistema, digamos que hay programadores, otro es el analista de calidad y el otro es un analista de base de datos. Incluso dependiendo el tamaño del software somos distribuidos en equipos y puede haber muchos equipos trabajando en un solo sistema.

[00:57] De esta forma cada equipo es responsable digamos por el mantenimiento de la base de datos, otro equipo es responsable por el formulario, otro equipo es responsable por cada uno de los botones de acá arriba y puede seguir así sucesivamente. ¿Pero cómo era hace diez años? No era para nada como es ahora.

[01:19] De hecho era muy raro ver tantos equipos en un solo proyecto de desarrollo de software, incluso era muy raro ver más de una persona trabajando en un formulario por así decirle, un sistemita. Incluso los sistemas de antes no eran así de amigables como lo vemos ahora y digamos con tan pocos campos, tan relacionados con lo que ahora llamamos the user experience.

[01:48] Antes eran ese tipo de formularios con un montón de campos y muchas veces teníamos que repetir esos campos en otros formularios también. Entonces, vamos a ver esto con un ejemplo en práctica. Supongamos que yo tengo una tienda y en mi tienda yo voy a dar promociones a cada cliente dependiendo no sé, si me compra cinco veces seguidas.

[02:18] Para eso yo voy a guardar el número de documento del cliente y la fecha en la que me está comprando. Cuando yo presione el botón guardar, básicamente aquí yo he escrito un pequeño pseudo código. Lo que yo voy a hacer es obtener una variable documento que va a ser el input de aquí, una variable fecha que va a ser obtenida de aquí y voy a guardar.

[02:42] Perfecto, no hay nada del otro mundo hasta aquí, está en lenguaje entendible para todos. Pero ahora llegan los conocidos corner case. ¿Qué sucede si en documento que hasta ahora es solo un campo de texto, yo coloco letras, coloco caracteres especiales? En la mayoría de casos los números de documento son solo números.

[03:11] Entonces, si yo coloco letras o algún carácter que no esté permitido, yo podría tener algunos problemas. Para eso yo podría tranquilamente implementar un método validar documento. Entonces aquí en mi tipo booleano yo voy a preguntar: "¿el documento es válido?" Voy a crear aquí un método para validar el documento, que puede ser por ejemplo evaluar si todos son números o si el número está siguiendo cierto patrón, etcétera. Va a depender mucho del país.

[03:45] Y recién si el documento es válido, que sería aquí, yo voy a proceder a guardar ese documento. Incluso aquí yo podría poner una cláusula else, mostrar algún mensaje de error. ¿Qué sucede si ahora yo no necesito digamos solamente guardar datos de mis clientes sino también datos de mis trabajadores? Entonces voy a crear otro formulario con número de documento, nombre de mi trabajador y el cargo que él ocupa.

[04:16] De la misma forma anterior que yo voy a obtener el documento, voy a obtener el nombre y el cargo, aquí hubo un pequeño error. Y nuevamente voy a validar el documento. Voy a escribir el código para validar el documento. Voy a hacer la misma pregunta, si el código es válido, entonces voy a guardar. Ahora, supongamos que yo necesito ahora, como ya guardé la información, necesito obtenerla, necesito ser capaz de visualizar esa información en pantalla.

[04:44] Voy a crear otro pequeño formulario. No va a ser formulario, va a ser una caja de texto básicamente, en el cual yo pongo el número de documento y doy a buscar. Nuevamente tenemos el mismo escenario. Necesito validar que aquel documento sea válido. Ustedes ven que estamos repitiendo la misma lógica para evaluar si el documento es válido o no en tres escenarios distintos.

[05:11] Entonces, este tipo de escenarios en el que tú comienzas a ver que estás repitiendo el mismo código, repitiendo la misma lógica en más de un campo, es un claro indicio de que necesitas abstraer eso en un único punto de acceso. De esta forma, si nosotros abstraemos fuera aquel método es válido, él va a ser accesible desde todos nuestros formularios y eso va a dar dos ventajas.

[05:46] La primera, si nosotros necesitamos optimizar ese método digamos, cambiar la lógica o algo, afectamos solamente nuestro método es válido y dejamos en funcionamiento formularios normal, igual que siempre y viceversa, la segunda razón es eso. Si necesitamos cambiar el comportamiento de nuestros formularios, no tenemos necesidad de intervenir en la validación del número del documento.

[06:15] Este es el principal objetivo de orientar a objetos. Separar responsabilidades y hacer el código reutilizable y entendible para cualquier programador.